Neural Networks for Background Audio Noise Suppression

36320, 38069, 35190

April 29, 2022

Abstract

Since the COVID-19 pandemic started, there has been a dizzying growth in telematic communication. Background noise in the age of virtual correspondence, is the main problem for business meetings and online classes via zoom and teams, producing serious distortions and difficulties in proper communication. In order to address this problem, our study compares the performance of three state of the art deep learning methods: Convolutional Recurrent Neural Networks (CRNNs), Convolutional Denoising Autoencoders (CNN-DAE) and Conditional Generative Adversarial Networks (CGANs). These models were developed as an alternative to traditional Dense Recurrent Neural Networks (DRNN). Finally, the study analyses a novel idea of reversing noise suppression and speech enhancement applications, by formulating models designed to add noise to clean speech. This could result in many applications in industry, such as film-making and sound engineering, where there is need for the addition of artificial sound. Following our analysis on a pre-constructed data set, we proceeded to enrich and expand it using a larger variety of sounds and clean speech, to aid in achieving higher performing models. Based on the results, it is clear that CNN-DAEs work best in suppressing noise and that they have potential when applied to the reverse use case.

Audio Noise Supression, Speech Enhancement, Audio Noise Addition, Convolutional Recurrent Neural Networks (CRNNs), Recursive Neural Networks (RNNs), Convolutional Denoising Autoencoders (CNN-DAE), Conditional Generative Adversarial Networks (CGANs).

1 Introduction

Background noises are the undesired sounds that appear accidentally in clean speech and impede the listener from having a pleasant hearing. These noises may be from an outside source such as traffic or construction activities, or from an inside source such as the TV or people conversing. The proper removal of acoustic distortions has always been important (Xu et al., 2014) but has become more prominent following the COVID-19 pandemic era, when most face-to-face activities became remote. The success of remote work and online classes depends essentially on the absence of background noise.

The technique of reducing the audio background noise is called speech enhancement (Pascual et al., 2017). Many speech enhancement techniques have been developed over the years, such as the Wiener filter (Chen et al., 2006). More recently, Neural Networks have been applied in this domain, with the use of Dense Neural Networks (DNN) and Deep Recurrent Neural Networks (DRNN) as they are shown to give optimal results when modelling sequential data (Xu et al., 2014). Although RNNs seem to be the most intuitive ones for speech enhancement, new approaches have been developed recently that have the potential of minimising background noise effectively. In this paper, three of these more recent architectures are developed and compared in order to determine if they are useful for solving the problem under investigation and which one shows the greatest potential.

The models implemented are Convolutional Recurrent Neural Networks (CRNN), Convolutional Denoising Autoencoders (CNN-DAE) and Conditional Generative Adversarial Networks (CGANs). The CRNN models combine two outstanding architectures, Convolutional Neural Networks which are able to extract high level features invariant to temporal changes, and Recurrent Neural Networks which are able to learn the longer term temporal context in the signals (Cakır et al., 2017; Tan and Wang, 2018). The second proposed approach, namely CNN-DAE, learns the main features of the data through encoding and decoding, and by training the model to disregard noise (Lu et al., 2013). They have proved themselves useful in image denoising (Bajaj et al., 2020; Gondara, 2016), which motivated the investigation of their application within the audio denoising context. Similarly, the employment of CGAN models in speech enhancement has been on the rise recently, with first being introduced in 2017, after having been proved to be useful in image generation and image denoising (Pascual et al., 2017). The CNN-DAE

and CGAN architectures have the additional advantage of having a lower computational complexity than the CRNN.

As a preliminary investigation and a proposal for further research, the prospect of adding noise to clean speech files through the employment of deep neural networks is analysed, a use-case which has yet to be explored in literature. To tackle this problem, the best model out of the three aforementioned models will be employed, and its performance of generating noisy speech files will be assessed.

Along with this report there are three Jupyter notebooks. The recommendation is to first read the report and then the notebooks given in the order below.

1. Noisy_Speech_Generation - A notebook to combine clean speech files with noise files to generate noisy speech files. Feature engineering is applied to the files including STFT with a Hamming window, log-power magnitude, and normalisation (refer to Section 2.2 and Section 2.3), to convert the audio files in to wave-forms/ arrays. These are then added to the pre-constructed data, and are used to train the models in the remaining two notebooks.
2. CRNN - A notebook containing the training and evaluation of Convolutional Recurrent Neural Networks.
3. CNN-DAE_CGANS_REVERSE - A notebook including training and evaluation of the Convolutional Denoising Autoencoders, Conditional Generative Adversarial Networks, and the Reverse Model.

2 Data

This section is divided into three parts. The first section provides a description of the pre-constructed data-set of noisy and clean speech files obtained from Kaggle. The second part, explains the generation of new and original audio files, through the addition of a diverse range of noises to clean speech signals. The final part focuses on the data feature engineering methods applied to all audio files, to produce waveforms of such form, that permit the models to extract the most important features and produce better results during training and evaluation. In this experiment, the two data-sets (pre-constructed and generated) are concatenated, and are employed in the analysis. In total there are 21,172 samples and the data was divided into training, validation and testing, with the following ratios: 90% training (17,994 samples) and 10% testing (2,119 samples), where 6% (1,059 samples) of the training data was used for validation.

2.1 Initial Data

The first data-set taken into consideration is the Libri Speech Noise Dataset, available at https://www.kaggle.com/datasets/earth16/libri-speech-noise-dataset?select=train.7z., which consists of 7,105 one-second waveform noisy speech audio files and their corresponding 7,105 clean counterparts, conforming a total of 14,210 audio files. The noises are taken from https://www.freesoundeffects.com/, which include ambient sound effects like city sounds, and the speech is taken from https://openslr.org/83/, a corpus containing male and female recordings of English from various dialects of the UK and Ireland. Each file within the data-set is given as a 3D array of dimensions (257,52,3), representing the number of samples per sequence, the length of sequences and the number of channels.

2.2 Data Generation

As mentioned previously, new data is generated by combining audio from two different sources. The clean speech data is collected from fluent-speech-corpus https://www.kaggle.com/datasets/tommyngx/fluent-speech-corpus, which was developed through crowdsourcing and contains 97 speakers saying 248 different phrases each. The noise data is obtained from FSDKaggle2019 https://www.kaggle.com/competitions/freesound-audio-tagging-2019/data and covers a wide range of sounds, including musical instruments, percussion, water, respiratory sounds, human voice, animals, and domestic sounds.

In signal processing, sampling is defined as the process of converting a signal from continuous time to discrete time, and sampling rate refers to the average number of samples obtained in the process of sampling (Vetterli et al., 2002). Before overlapping the separate audio files, we need to match the sampling rate of the files to ensure that they are on the same time-base (play at the same speed) and to provide a better audio quality. First, downsampling (lowering the sampling rate of a digital signal audio) (Sonnaillon and Bonetto, 2005) is applied to the speech data (here the sampling rate is 16 kHz which means 16,000 samples per second). The noise signals

are then re-sampled to the same rate. A few seconds are then added to the noise to ensure its length is greater than that of the maximum length of the speech audio files in our data-set. Finally, the processed speech audios files and noise files are merged and the length of the combined generated audios are cut to one second, to remain consistent with the files within the Libri Speech Noise Dataset. Figure 1 and Figure 2, represent the spectrograms (plot of amplitude over time) and oscillograms (plot of frequency over time) of clean speech and the corresponding generated noisy speech. As can be seen from both the oscillograms and spectrograms, noise has been successfully added to the entire clean signal, increasing the amplitude and fluctuations observed, particularly in the first few deciseconds (Ferguson and Kewley-Port, 2007).

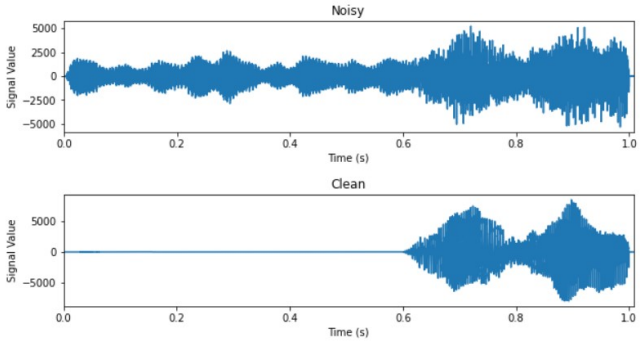


Figure 1: Oscillograms of Generated Noisy and Clean Speech signal

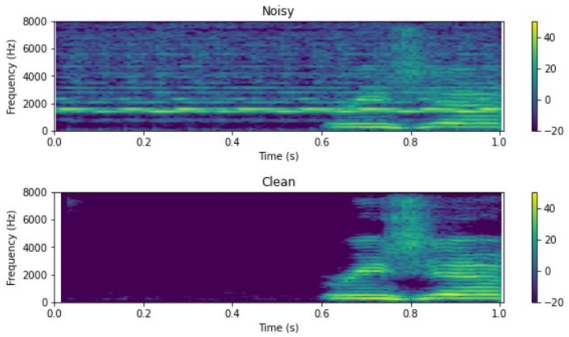


Figure 2: Spectrograms of Generated Noisy and Clean Speech signal

2.3 Data Engineering

One common method to isolate a certain desired signal from a chaotic signal, is to apply Fourier transform, which identifies individual signals by frequency (Harris, 1978). Fourier transform decomposes a signal in a time domain representation, into an alternate representation of sine and cosine functions with different frequencies (Storey, 2002). Discrete Fourier transform (DFT) is applied when the waveform has discrete values (Harris, 1978). A more practically developed algorithm of DFT is the fast Fourier transform (FFT), which provides rapid computation of the decomposition of waveforms (Thakur and Mehra, 2016).

In this study, speech is a time-varying signal (Hall et al., 1983), i.e., it has complicated properties changing over time. If we apply orthodox Fourier transformation to a spoken sentence [1], the spectrum obtained is an average of all phonemes in the utterance (Kluender et al., 2013). The properties and spectra of the speech can be attenuated by the background noise, which makes machine learning less accurate (Schroeder, 2004). So, some additional analysis is required to see the spectrum of each phoneme and to reflect the characteristics of the voice (Schroeder, 2004). To do this, one can split the signal into short segments which include only one phoneme to obtain the separated properties (Allen and Rabiner, 1977). In this case, the characteristic of the sentence is not altered in each segment. Such segmentation of the signal is called windowing. By windowing and employing DFT to each segment, one derives short-time Fourier transform (STFT) of the signal (Allen and Rabiner, 1977). The short-time Fourier transform of a function $f \in L^2(\mathbb{R})$ given a window $g \in L^2(\mathbb{R})$, at frequency ω and time shift ℓ is defined as (Allen and Rabiner, 1977):

$$STFTf(\ell, \omega) \coloneqq \mathcal{F}[f(\cdot)g((\cdot) - \ell)](\omega) = \int_{-\infty}^{\infty} f(t)g(t - \ell)e^{-2\pi i \omega t} dt \tag{1}$$

When windowing, one can use different smooth functions to manage the discontinuities at the knots between two segments. A common window function in signal processing is Hanning window (Blackman and Tukey, 1958), which is defined as below in our experiment:

$$\omega(n) = 0.5 - 0.5\cos\big(\frac{2\pi n}{M-1}\big),\ 0 \leq n \leq M-1 \tag{2}$$

Hanning window usually presents good frequency resolution and reduced spectral leakage (Pielawski and Wählby, 2020). The input signal can be recovered from STFT which is called phase retrieval (Pielawski and Wählby, 2020). So in this experiment, we apply STFT to the data-set mentioned above using Hanning window. A multiple of the base-10 logarithm of the produced arrays following feature engineering is calculated, and the resulting arrays are then re-scaled by dividing by 255, to attain values between 0 and 1.

[1] A spoken sentence is a sequence of phonemes.

3 Methodology

In this section, the formulation and theoretical aspects of each model applied, are explained in detail. Furthermore, the explanation of model parameters employed within the architecture of the models can be found in Section 4.

3.1 CRNNs

The Convolutional Recurrent Neural Network (CRNN) combines Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). In previous studies, CNNs have been applied in many tasks, like speech recognition (Graves et al., 2013), while RNNs were designed to tackle sequence problems, which confirms the dependency of the current output of series data on previous data (Elman, 1990). Based on previous work, CNNs can capture higher level features which keep stable with the change of local spectral shifts, while RNNs present good results for long-time learning (Cakır et al., 2017). One of the common applied RNNs is the long short-term memory (LSTM) network (Hochreiter and Schmidhuber, 1997), which is used to obtain long-term properties for series data. The authors Cho et al. (2014), employed LSTM for machine translation. The combination of CNNs and RNNs was first proposed by (Tang et al., 2015). It has been found to be a powerful method in many fields, such as hyperspectral data classification (Wu and Prasad, 2017), music classification (Choi et al., 2017) and traffic forecasting (Li et al., 2017). Furthermore, CRNNs are also applied in audio signals (Cakır et al., 2017). In this study, we propose a CRNN for speech enhancement.

The convolutional layers and LSTM layers used in the proposed CRNN will be introduced explicitly. The filtered audio sequences go through the stacked convolutional layers which operate as below, which is defined as temporal convolution z (Li et al., 2019).

$$z[m] = \sum_{i=-l}^{l} x[i] \cdot \delta[m-i] \tag{3}$$

The x in this operation denotes the input signal and δ represents the kernel (Li et al., 2019). Through this convolutional operation, the model can capture the signal's spatial characteristics. With great memory power, the LSTM may extract time-variant properties of the input signal, which is good for learning long-term dependencies (Bengio et al., 1994). Specifically, the audio characteristics of each moment can be obtained through LSTM units, where the computation processes are shown below (Tan and Wang, 2018):

$$f_t = \sigma(W_f \cdot x_t + U_f \cdot h_{t-1}) \tag{4}$$
$$i_t = \sigma(W_i \cdot x_t + U_i \cdot h_{t-1}) \tag{5}$$
$$\tilde{c}_t = tanh(W_c \cdot x_t + U_c \cdot h_{t-1}) \tag{6}$$
$$o_t = \sigma(W_o \cdot x_t + U_o \cdot h_{t-1}) \tag{7}$$

Where the activation function σ is the logistic sigmoid function and $tanh$ is the hyperbolic tangent function, x_t, $\tilde{c}_t$, c_t, and h_t denote input, block input, memory cell and hidden activation at time t accordingly. The variables W and U represent weight and bias matrices (Tan and Wang, 2018). With these formulas, memory cell and final output h_t can be calculated by (Wu and Prasad, 2017):

$$c_t = i_t \odot \tilde{c}_t + f_t \odot c_{t-1} \tag{8}$$
$$h_t = o_t \odot tanh(c_t) \tag{9}$$

In this case, $\odot$ denotes the element-wise multiplication (Wu and Prasad, 2017). These functions demonstrate that LSTM can control the flow of information, thus acquiring long-term dependencies more easily (Wang et al., 2019).

3.2 Convolutional Denoising Autoencoder (CNN-DAE)

The structure of an autoencoder encapsulates a model consisting of an encoder and a decoder. Traditionally, it takes an input x, maps it to a latent representation y by encoding it whilst preserving the most important features, and then decodes y to obtain a reconstruction z from the compressed representation of the original input (Vincent et al., 2008; Kovenko and Bogach, 2020).

When applying autoencoders for noise reduction, an extension of the classical autoencoder, denoising autoencoders, first introduced by Vincent et al. (2008), may be applied. As a modification to the traditional approach, the

Denoising Autoencoder (DAE) is trained to reconstruct "a clean "repaired" input from a corrupted, partially destroyed one". The input to the model in our case is thus the noisy speech audio and the target is the clean speech audio. Mathematically, combinations of clean and noisy signals (x,$\tilde{x}$) are used to estimate $p(x|\tilde{x}) = p_D(x|h)$ where h represents the encoded latent representation, and the DAE is trained to minimise the loss function (Goodfellow et al., 2016; Alain and Bengio, 2014; Bengio et al., 2013):

$$L_{DAE}(x) = E_{\tilde{x}\sim C(\cdot|x)}[L(x, g(f(\tilde{x}))]$$

Whereby $\tilde{x} \sim C(\cdot|x)$, f refers to the encoder and g is the decoder (Goodfellow et al., 2016; Alain and Bengio, 2014; Bengio et al., 2013).

Given the representation of our clean and noisy data, as 3-dimensional arrays in the frequency time domain followed by feature engineering, convolutional autoencoders can also be considered ideal. These are autoencoders consisting of convolutional layers in which local spatiality is preserved as weights are shared among all input locations (Fu et al., 2016). A convolutional neural network structure has been seen to be ideal in applications of speech denoising as described in Fu et al. (2016), as it can deal with local temporal-spectral structures of speech signals, allowing for a separation between the speech and noise within a given input signal. The use of a convolutional denoising autoencoder is thus proposed to effectively tackle our noise suppression and speech enhancement task.

3.3 Conditional GANs

Generative Networks (GANs) are a type of neural architecture which employs two types of models which compete against each other, the generator and the discriminator (Pascual et al., 2017). The generator captures the most important features of the data, while the discriminator learns the decision boundary to classify the data into two classes (Baby, 2020). The generator is used to create fake data which is then passed to the discriminator which classifies it as real or fake (Michelsanti and Tan, 2017). If the discriminator cannot distinguish the fake samples from the real ones up to a specified threshold, then the GAN is considered to be a good model (Hitaj et al., 2017). In this paper, we decided to focus on Conditional GANs as they are the ones that best approximate the given problem. Conditional GANs are very similar to the GAN models, however, in the former, the discriminator and generator are conditioned on some extra information.

In a more mathematical way, let x be a sample of real data where $x \in \mathbb{R}^d$ and $d \in \mathbb{R}$ and let y be a random variable (Donahue et al., 2018; Pascual et al., 2017). The generator is a neural network that takes as input a random noise z and generates a data sample $\tilde{x} = G_\phi(z, y)$, where ϕ are the parameters that need to be trained (Donahue et al., 2018). The fake data is passed through the discriminator along with the real data, and it outputs the prediction probability of the sample being real. Hence, assume $D_\theta(x, y) \in [0, 1]$, where θ are the parameters that need to be specified (Donahue et al., 2018). While the generator will try to maximise the expected log-likelihood of incorrectly classifying a fake sample, the discriminator will minimise this maximum loss (Michelsanti and Tan, 2017). Hence, the generator and discriminator will play the following zero-sum game (Baby, 2020; Michelsanti and Tan, 2017):

$$\min_\theta \max_\phi (-E_{x,y\sim p(x,y)}[log(D_\theta(x, y))] - E_{z\sim p_z, y\sim p(y)}[log(1 - D_\theta(G_\phi(z, y), y)])$$

The solution will be a trade-off between the generator and the discriminator (Pascual et al., 2017). The optimal discriminator will be the one stated in Theorem 1 below (Arjovsky and Bottou, 2017).

Theorem 1. *Assume that $D_\theta(x) \in [0, 1]$ for any x and some θ. Furthermore, p and q are the distribution of the real and fake data, respectively. Then, for any fixed generator $G_\phi(z)$, the optimal discriminator is defined as follows:*

$$D_\theta(x, y) = \frac{p(x, y)}{p(x, y) + q(x, y)}$$

Theorem 2. *$\exists$ a solution to the minimax problem $\iff$ $q = p$. Where p is the distribution of the real data and q is the one of the fake data outputted by the generator.*

Furthermore, Theorem 2 defined above implies that the generator G_ϕ minimises the Jensen-Shannon divergence defined as follows (Arjovsky and Bottou, 2017):

$$D_{JS}(p, q) = \frac{1}{2} D_{KL}(p, \frac{p+q}{2}) + \frac{1}{2}(q, \frac{p+q}{2})$$

Here, $D_{KL}(p,q) = \int log(\frac{p(x,y)}{q(x,y)})p(x,y)d\mu(x,y)$ is the Kullback-Leibler(KL) divergence and p and q are densities with respect to some measure μ on $\mathcal{X}$ (Arjovsky and Bottou, 2017).

In our case, the noisy sample, y, will be passed through the generator without the extra noise, z. This is due to the fact that, as stated by Isola et al. (2017), the technique of passing extra noise to the input, is not effective, given our interest to achieve the most accurate mapping between the noisy (y) and clean samples (x), and not to learn the distribution of the clean samples. Then, the enhanced or fake observation ($\tilde{x}$) and the real one (x) are passed through the discriminator, being both conditioned on the noisy sample (y) (Michelsanti and Tan, 2017). Finally, as explained above, the discriminator will try to determine the probability of the samples being fake.

4 Model Parameters

Given the fact that all three models are a particular formulation of a neural network, they share many model parameters. The different types of layers, hyperparameters, loss functions and optimisers used within these architectures, are thus defined.

4.0.1 Layers

- *Convolutional 2D:* Layer that uses kernel filters of a defined size (smaller than that of the input) to compute the convolution operation of given inputs and extract fundamental features (Basant Agarwal, 2020). Each filter is "convolved with the input volume to compute an activation map made of neurons" (Marco Leo, 2018), which is 2 dimensional, hence the name.
- *Convolutional 2d Transpose/ Deconvolution:* Layer that use kernel filters which define the convolution. However, they operate by swapping the forward and backward passes of the convolution (Dumoulin and Visin, 2016).
- *Batch Normalisation:* Layer that has the effect of reducing internal covariate shift, by fixing the variances and means of the inputs of a layer (Sergey Ioffe, 2015). This has the effect of decreasing training times and regularizing the model, thus reducing the need for dropout layers, as well as allowing the use of higher learning rates by mitigating the risk of the model diverging from the minimum (Masters and Luschi, 2018). Training performance is thus significantly improved (Masters and Luschi, 2018; Sergey Ioffe, 2015).
- *LSTM:* Layer that consists of a set of given hidden nodes. It learns long-term dependencies between time steps and performs additive interactions (Salman et al., 2018).
- *Dropout:* Layer that randomly drops units and their connections from the neural network during training (Srivastava et al., 2014). Used to prevent overfitting.
- *Reshape:* Layer that reshapes the inputs into a given shape.
- *Permute:* Layer that permutes the dimensions of the input following a set order.

4.0.2 Hyperparameters

- *Batch Size:* The batch size specifies the number of observations that the algorithm will go through before updating the parameters of the model (Brownlee, 2018).
- *Number of epochs:* The Number of epochs, determines how many times the algorithm will go through the entire data set before updating the parameters when training, and consists of one or more batches (Brownlee, 2018).
- *Number of nodes in a layer:* This hyperparameter is known as the width of the layer (Karsoliya, 2012).

4.0.3 Loss Functions

The main aim of supervised neural networks is to minimise a predefined loss function during the training process.

- *MSE:* The MSE Loss function can be defined as follows, where in our case y, $\hat{y}$, and N represent the target clean signal, predicted signal, and the size of the sample, respectively (Park, 2022).

$$L_{MSE}(y, \hat{y}) = \frac{1}{n}\sum_{n=1}^{N}(y - \hat{y})^2$$

- *Binary Cross-entropy:* It is defined as follows, where $y = \{0, 1\}$ is a binary indicator and p is the predicted probability of an observation x belonging to class 1 (Creswell et al., 2017). In our case, p will be the probability of a sample being the real clean speech, $y = 1$, and $1 - p$ the probability of being the fake enhanced speech. It is defined as follows (Creswell et al., 2017):

$$L(y, p) = -(ylog(p) + (1 - y)log(1 - p))$$

4.0.4 Activation Functions

- *Rectified Linear Units (ReLU):* ReLU activation function works by thresholding the value at 0 (Agarap, 2018). In other words, given an input x, it produces a 0 if x is negative and produces a linear function for x greater or equal to 0. This is mathematically defined as $f(x) = max(0, x)$ (Agarap, 2018). ReLU may outperform other activations such as sigmoid and tanh, due to the fact that ReLU is unsaturated and thus does not suffer from gradient diffusion problems (Wang et al., 2020).
- *Leaky Rectified Linear Units (Leaky ReLU):* Activation function similar to the ReLU activation function but if $x < 0$, instead of 0 it takes a small negative number (Xu et al., 2020). This resulting value depends on the coefficient of leakage determined by the user (Xu et al., 2020). It is thus defined as $\max(\alpha x, x)$, where $\alpha \in [0, 1]$ is the coefficient of leakage (Xu et al., 2020).
- *Parametric Rectified Linear Units (PReLU):* Activation function which takes the idea of the Leaky ReLU activation further, by having the coefficient of leakage as a parameter that is trained along with the other neural network parameters (QingJie and WenBin, 2017). Hence, it is defined as $f(x_i) = x_i$ for $x_i \geq 0$ and $f(x_i) = a_i x_i$ for $x_i < 0$ (QingJie and WenBin, 2017).
- *Sigmoid:* The Sigmoid Activation function is a non-linear activation function, also known as the logistic function. It produces values between 0 and 1 and is given by $f(x) = \frac{1}{(1+e^{-x})}$ (Wang et al., 2020; Nwankpa et al., 2020).

4.0.5 Optimisers

To optimise the loss function, neural networks make use of optimisers. In this case the Adam Optimiser is considered.

- *Adam Optimiser:* The Adam optimiser, the name being derived from adaptive moment estimation, is a method which combines the advantages AdaGrad and RMSProp methods (Kingma and Ba, 2014a), and operates efficiently when working with a lot of parameters or data (Klooster, 2021). It is able to outperform other optimisers (Klooster, 2021; Metz et al., 2018), with benefits including step-sizes being approximately bounded, its capability to operate with sparse gradients and the fact that it does not require a stationary object (Kingma and Ba, 2014a).

5 Metrics

Below are all the metrics considered when evaluating our noise suppression models. The metrics of the noisy data when compared to the clean data are first computed as a baseline reference. The metrics of the predicted signals when compared to the clean signals are also computed for each model. These achieved scores can be visualised in Table 5. Please note that for all metrics the higher the value, the better the performance of the model.

5.1 BSS-Eval Scores

The BSS-Eval Scores refers to a toolbox of metrics originally proposed by Vincent et al. (2006), to measure the performance of various source separation algorithms in an evaluation framework. The main intuition behind these metrics, lies in the fact that given an estimate $\hat{s}(t)$ of a source $s_j(t)$, one could decompose the given estimate as (Févotte et al., 2005):

$$\hat{s}(t) = s_{target}(t) + e_{interf}(t) + e_{noise}(t) + e_{artif}(t)$$

The variables of the equation above are defined as follows (Févotte et al., 2005):

$s_{target}(t)$ - allowed distortion of the target source $s_j(t)$

$e_{interf}(t)$ - allowed deformation of the sources dut to interferenc of unwanted sources

$e_{noise}(t)$ - allowed deformation of the perturbing noise

$e_{artif}(t)$ - prohibited deformations or artifacts introduced by the separation algorithm.

Mathematically, these can be defined in the following way (Dumoulin and Visin, 2016):

$$\begin{aligned} s_{\text{target}} &:= P_{s_j}\widehat{s}_j, \\ e_{\text{interf}} &:= P_{\mathbf{s}}\widehat{s}_j - P_{s_j}\widehat{s}_j, \\ e_{\text{noise}} &:= P_{\mathbf{s},\mathbf{n}}\widehat{s}_j - P_{\mathbf{s}}\widehat{s}_j, \\ e_{\text{artif}} &:= \widehat{s}_j - P_{\mathbf{s},\mathbf{n}}\widehat{s}_j. \end{aligned}$$

$$\begin{aligned} P_{s_j} &:= \Pi\{s_j\}, \\ P_{\mathbf{s}} &:= \Pi\{(s_{j'})_{1\leq j'\leq n}\}, \\ P_{\mathbf{s},\mathbf{n}} &:= \Pi\{(s_{j'})_{1\leq j'\leq n}, (n_i)_{1\leq i\leq m}\}. \end{aligned}$$

Where $\Pi\{y_1.y_2, ...y_k\}$ denotes the orthogonal projector onto the space spanned by $y_1.y_2, ...y_k$, and where $s_{j'}$ is the unwanted source and s_j is the wanted source (Dumoulin and Visin, 2016). Based on this decomposition, one can define several metrics including Source-to-Distortion Ratio (SDR), Scale Invariant SDR (SI-SDR), Source-to-Artifact Ratio (SAR) and Source Image to Spatial distortion Ratio (ISR).

5.1.1 Source-to-Distortion Ratio (SDR)

The Source-to-Distortion Ratio or SDR, which captures the overall separation quality of the algorithm (Venkataramani et al., 2018), can be defined as:

$$SDR = 10 * log_{10} \frac{\|s_{target}\|^2}{\|e_{interf} + e_{noise} + e_{artif}\|^2}$$

It takes into account all three distinct error components representing spatial (or filtering) noise, interference and artefacts (Vincent et al., 2007; Ivry et al., 2021). Furthermore, it does not make a distinction between them (Vincent et al., 2007; Ivry et al., 2021).

5.1.2 Scale Invariant SDR (SI-SDR)

The SI-SDR or otherwise scale invariant SDR, is a modified version of the SDR proposed by Le Roux et al. (2019). Its main purpose is to combat artificially inflated SDR scores, due to SDR's dependency on the amplitude of the signal. Although SI-SDR is not sensitive to amplitude scaling, it is a quicker computation as it does not require windowing the estimated and ground truth signals like SDR. It is defined as (Le Roux et al., 2019):

$$SI\text{-}SDR = 10 * log_{10} \frac{\|s_{target}\|^2}{\|e_{res}\|^2}$$

In this case, the estimate is decomposed as $\hat{s} = s_{target} + e_{res}$ and $s_{target} = \alpha s$ is the scaled reference with optimal scaling factor α (Le Roux et al., 2019).

5.1.3 Source-to-Artifact Ratio (SAR)

Source-to-Artifact Ratio is generally interpreted as the amount of unwanted artifacts a source estimate has with relation to the true source. It can be defined in the following way (Vincent et al., 2006):

$$SAR = 10 * log_{10} \frac{\|s_{target} + e_{interf} + e_{noise}\|^2}{\|e_{artif}\|^2}$$

When training a noise separation or suppression network, the model may introduce additional sounds in the results, non-existent in the original signals. SAR therefore allows for the distinction of estimation errors dominated by such artefacts and is therefore a measure of the ability of the model to perform well without introducing additional interference (Venkataramani et al., 2018; Vincent et al., 2006).

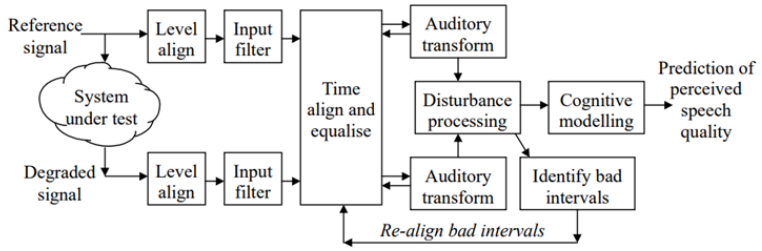
5.1.4 Source Image to Spatial distortion Ratio (ISR)

Source Image to Spatial distortion Ratio (ISR) is a metric focused on the level of spatial distortion or noise within the given signal. The formula to calculate the metric can be seen below (Vincent et al., 2007).

$$ISR = 10 * log_{10} \frac{\|s_{target}\|^2}{\|e_{noise}\|^2}$$

5.2 Perceptual Evaluation of Speech Quality (PESQ)

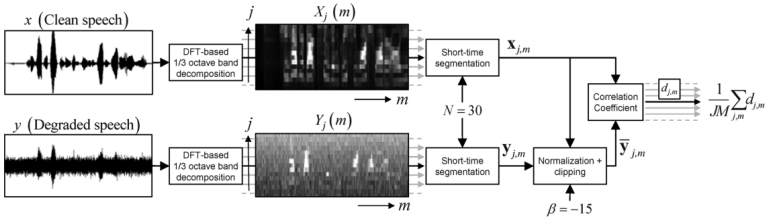
The PESQ score proposed by Rix et al. (2001), an enhanced version of PSQM, is a model assessing speech quality of the following structure:



In PESQ, the original and degraded signals are time aligned and equalised, and then mapped to an internal representation using a perceptual model (Rix et al., 2001). Distortion parameters are extracted from the difference in these representations, and are used by the cognitive model to give a prediction of the degraded signal's perceived speech quality and subjective mean opinion score (MOS) (Beerends et al., 2002; Rix et al., 2001). The PESQ score ranges between between -0.5 and 4.5 (Leglaive et al., 2020).

5.3 Short-Time Objective Intelligibility (STOI)

The STOI metric, proposed by Taal et al. (2011), is "a function of a TF-dependent intermediate intelligibility measure", whose structure can be seen below.



It operates by firstly decomposing the clean and degraded speech into DFT-based bands, followed by comparison of normalised and clipped short time temporal envelopes of the two sources, using a correlation coefficient (Taal et al., 2011). The average of short-time intermediate intelligibility measures across all bands j and all frames m, are averaged in order to obtain a final STOI score (Lightburn and Brookes, 2016). The score ranges between 0 and 1 (Leglaive et al., 2020).

5.4 MOSNet

MOSNet refers to a neural network built to predict a Mean Opinion Score (MOS) based on human ratings, of the quality and similarity of enhanced speech (Lo et al., 2019). In contract to relative scores, MOSNet is correlated with human perception and is a form of non-intrusive audio assessment metrics given the fact that the clean signal is not needed for reference. To create ground truth scores and train the model, 120,000 evaluations were conducted in which listeners evaluated the speaker similarity and naturalness of the given sample by submitting a MOS rating, with 1 being the lowest and 5 the highest (Lo et al., 2019). The model is then formulated as a regression task and different model architectures can be chosen, as seen from Figure 3 (Lo et al., 2019).

model	**BLSTM**	**CNN**	**CNN-BLSTM**
input layer	input (***N X 257 mag spectrogram***)		
conv. layer		$\left\{\begin{matrix} conv3-(channels)/1 \\ conv3-(channels)/1 \\ conv3-(channels)/3 \end{matrix}\right\} X4$ $channels$ = [16, 32, 64, 128]	
recurrent layer	BLSTM-128		BLSTM-128
FC layer	FC-64, ReLU, dropout	FC-64, ReLU, dropout	FC-128, ReLU, dropout
	FC-1 (***frame-wise scores***)		
output layer	average pool (***utterance score***)		

Figure 3: Configuration of different model MOSNet architectures. The convolutional layer parameters are denoted as conv{receptive field size}-{number of channels}/{stride}. The ReLU activation function after each convolutional layer is not shown for brevity. N is for the number of frames.

In our specific case, the MOSNet metric is calculated on the basis of the CNN-BLSTM model architecture (Lo et al., 2019).

5.5 Speech-to-Reverberation Modulation energy Ratio (SRMR)

The SRMR metric was introduced by Falk and Chan (2008) as a non-intrusive measure of reverberant/noisy and dereverberated/clean speech quality. It uses the modulation spectral representation of the signals, as the modulation envelopes give useful insights into objective speech intelligibility and quality. The SRMR score is "computed as the ratio of the average modulation energy content (over all frames) in the first four modulation bands (3 - 20 Hz), to the average modulation energy content available in the last four modulation bands (20 - 160 Hz)." (Santos et al., 2014). It is given by (Falk and Chan, 2008):

$$SRMR = \frac{\sum_{k=1}^{4} \bar{\epsilon}_k}{\sum_{k=5}^{K^*} \bar{\epsilon}_k}$$

$$\bar{\epsilon}_{j,k} = \frac{1}{N_{act}} \sum_{i=1}^{N_{act}} \epsilon_{j,k}^{act}(i)$$

Where K^* is dependent on the speech signal and is based on "the bandwidth of the lowest gammatone filter for which 90% of the total energy is accounted for" (Falk and Chan, 2008), $\epsilon_{j,k}^{act}(i)$ is the modulation energy of active speech frames and N_{act} is their count.

6 Implementation and Evaluation of Main Models

Next, the implemented model setup for the CRNN, CNN-DAE and CGANs are presented, along with their respective results.

6.1 Convolutional Recurrent Neural Networks (CRNN)

6.1.1 Model Setup

The architecture of the proposed CRNN is shown in Figure 4. In this network, a zero padding layer and cropping layer are added as the first and last layers of the model, to maintain the original input size. Two 2-dimensional convolutional layers are used in the encoder with 128 and 256 units, while two 2-dimensional convolutional transpose layers are used in the decoder with 256 and 128 units. All these layers are associated with a 3×3 filter. Convolutional layers are employed to extract the main features of the input signals. ReLU is employed as the activation function and strides is set to (2,2) in these convolutional layers, which are then followed by batch normalisation layers. The batch normalisation layers standardise the inputs, which regulates the mean of inputs to approximately zero and ensures unit variance. A flatten layer and reshape layer are added to transform the output of the convolutional layers to the required input shape of the LSTM layers. The permute layer following the reshape layer, changes the direction of the axes of the feature vectors.

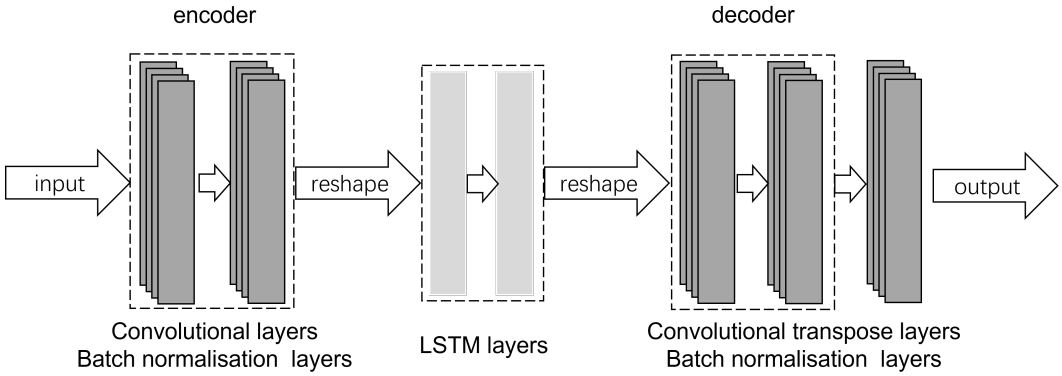


Figure 4: Architecture of proposed CRNN.

According to (Zia and Zahid, 2019), two LSTM layers are employed to make the model deeper. The LSTM layers are compatible with 2-dimensional feature vectors. After these, a permute layer and a reshape layer is used to pass the data to the decoder layers in proper shape. At last, another convolutional transpose layer with 3 units is added in the end with a sigmoid activation function, to convert the sequence back to the input shape and produce a final output. The model is trained with Adam optimiser (Kingma and Ba, 2014b) with a learning rate of 0.001. The mean squared error (MSE) is the loss function. The model is trained with batch size of 16 and for 25 epochs. The choice of the number of epochs is the optimal one due to the fact that after some experimentation, increasing the number of epochs leads to overfitting. Besides this final network, other models have been tested with changes such as the number of convolutional layers, the filter size of each convolutional layers and deconvolutional layers, and the number of LSTM layers. When a large number of convolutional layers are used in the network where the strides are set to (2,2) (the deconvolutional layers changes accordingly), the LSTM layers would require more hidden nodes. This resulted in insufficient storage space. Therefore the choice of the architecture is two convolutional layers and three deconvolutional layers. When we tried to apply only one LSTM layer, the loss of the model decreased very slowly resulting in a high complexity, which is non-viable in the long run. A detailed description of the final CRNN formulation is shown in Table 1.

Layer (type)	Output Shape	Parameters
zero_padding2d_2 (ZeroPadding2D)	(None, 260, 64, 3)	0
conv2d_4 (Conv2D)	(None, 130, 32, 128)	3584
batch_normalization_9 (BatchNormalization)	(None, 130, 32, 128)	512
conv2d_5 (Conv2D)	(None, 65, 16, 256)	295168
batch_normalization_10 (BatchNormalization)	(None, 65, 16, 256)	1024
flatten_2 (Flatten)	(None, 266240)	0
reshape_4 (Reshape)	(None, 256, 1040)	0
lstm_4 (LSTM)	(None, 256, 256)	1328128
lstm_5 (LSTM)	(None, 256)	525312
dropout_2 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 133120)	34211840
reshape_5 (Reshape)	(None, 65, 16, 128)	0
conv2d_transpose_5 (Conv2DTranspose)	(None, 130, 32, 256)	295168
batch_normalization_11 (BatchNormalization)	(None, 130, 32, 256)	1024
conv2d_transpose_6 (Conv2DTranspose)	(None, 260, 64, 128)	295040
batch_normalization_12 (BatchNormalization)	(None, 260, 64, 128)	512
conv2d_transpose_8 (Conv2DTranspose)	(None, 260, 64, 3)	387
batch_normalization_13 (BatchNormalization)	(None, 260, 64, 3)	12
cropping2d_2 (Cropping2D)	(None, 257, 62, 3)	0

Total params: 36,957,711

Trainable params: 36,956,169

Non-trainable params: 1,542

Table 1: Model network architecture of proposed CRNN.

6.1.2 Results

Figure 5 shows the training and validation loss of CRNN over training epochs. It is observed that the performance on the train data-set is good and improves over the epochs, which means that the network is learning (Emmert-Streib et al., 2020). Specifically, the training loss drops from 0.0817 to 0.0373 after the first epoch. It then decreases smoothly to approximately 0.0300 in following epochs. The loss of validation falls from 0.0391 to 0.0363 in the first epoch, and remains stable around 0.0336 with subtle fluctuations. Moreover, this validation loss diverges as the number of epochs increases, which shows a sign of potential overfitting. Given that the MSE is 0.0352 after assessing the network on the test data-set, the model is not generalising well to unseen data. Therefore the setting of dropout layers needs to be regulated to ensure generalisation. The PESQ and STOI scores are 1.40 and 0.64, respectively. The proposed CRNN does not denoise the speech audio very well, however, the scores are greater than those of the baseline noisy speech, indicating that the model extracts some of the properties of the speech audios during training.

A possible reason of the low performance could be that the CRNN model overfits the voice in the training data-set. To design a CRNN, one needs to pay attention to the trade-off between capturing more details and getting better generalisation (Liebl and Burghardt, 2020). To deal with overfitting, a larger training data-set is desired. Other potential solutions are to increase the dropout rate and shuffle the data-set for every epoch.

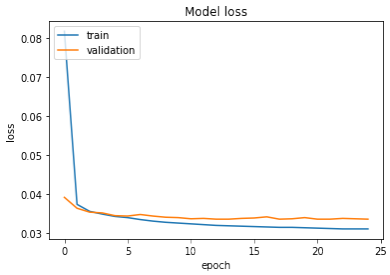


Figure 5: Training and validation loss of proposed CRNN.

6.2 Convolutional Denoising Autoencoder (CNN-DAE)

6.2.1 Model Setup

Following experimentation it has been observed that a relatively simple and not very deep structure of the model is preferred, due to the tendency of the model to overfit when moving to more complex structures. Cases of complex models were also observed, in which although training and validation loss of the model were very low, the noise metrics results were very poor. This is an indication that although the models were producing arrays similar to those of clean speech, the points at which the predicted and true arrays differed were of such nature (usually artefacts), that the quality of the predicted signal was significantly hindered. Furthermore, having tested the models with varying hyperparameters and layers including different batch sizes, number of epochs, learning rates, number of filters, kernel sizes, Dropout layers and activation functions including Leaky ReLU, the following observations were made.

1. A large number of filters within the Convolutional layer is beneficial to the results achieved by the model, however this has a cost of increasing training times significantly. This agrees with previous research of similar use cases in which large filter sizes were associated with lower cost function values (Mou and Li, 2021).
2. Sigmoid activation layer in the final convolutional layer (output layer) performed better than ReLU. This agrees with previous research in which ReLU is recommended to be used only in the hidden layers and not in the output later (Siddharth Sharma, 2020).
3. Results with small batch sizes were superior to those of large batch sizes, as perhaps the "possibility of using more up-to-date gradient information has a positive impact on training." (Masters and Luschi, 2018) and leads to more robust convergence.
4. Learning rates between the ranges 0.001 and 0.005 were seen to be the most effective. As in many other papers such as Wilson and Martinez (2001), large learning rates tend to overshoot a minimum leading to expensive backtracking. This indeed was the case when trialing large learning rates during CNN-DAE training.

The chosen structure can be visualised in Table 2. Both the encoder and the decoder consist of two Convolutional 2D and Convolutional 2D Transpose layers respectively, of 256 units associated with a 5x5 filter, ReLU activation function and strides set to (2,2). These layers are used for jointly convolving the inputs and finite impulse response filters, and down-sampling the output (Riad et al., 2022). Using strides as a down-sampling method was preferred over using max-pooling layers, due to the fact that max pooling is a fixed operation with no trainable parameters in comparison to down-sampling within the convolutional layer itself, which allows for learning important features. Usage of this down-sampling method can be seen within other model architectures such as ResNet and SqueeZNet. (Iandola et al., 2016; He et al., 2016)

Each Convolutional/Convolutional Transpose layer is also followed by a batch normalisation layer to standardise the inputs to a network, and hence reduce both the generalisation error (overfitting), as well as accelerate training. The final trainable layer of the model is a Convolutional 2D Transpose layer of 3 units and sigmoid activation function, in order to obtain a final predicted clean signal from the CNN-DAE. Encapsulating all these layers is a zero-padding and cropping layer, which increase the array size (by adding zeros) and decrease the array size respectively. These are used to maintain input and output shape during the auto-encoding process. To train the model, an MSE loss function was used in combination with the Adam Optimiser. A learning rate of 0.005 and batch size of 16 were employed, and the model was trained for 25 epochs.

6.2.2 Results

Following the implementation of the chosen network architecture as described in Section 6.2.1, below lie the testing and training results achieved. Let's first visualise the evolution of the training and validation loss over the course of 25 epochs of training. As is evident from Figure 6, both the training and validation loss decrease during training, indicating that the model is in fact learning and operating as expected (Emmert-Streib et al., 2020). The training loss decreases much more smoothly from a value of 0.032, down to a value of 0.0231. In contrast, the validation loss fluctuates and as expected is slightly more than that of the training, reaching as low as 0.0239. When evaluated on the test data, the MSE achieved takes a value of 0.0241 suggesting that the trained model performs well on unseen data and is generalisable.

To further assess the performance of CNN-DAE in noise suppression and speech enhancement, the aforementioned metrics (Section 5) are applied to compare the predicted and true clean test samples, the results of which can be

Layer (type)	Output Shape	Parameters
zero_padding2d (ZeroPadding2D)	(None, 260, 64, 3)	0
conv2d (Conv2D)	(None, 130, 32, 256)	19456
batch_normalization (BatchNormalization)	(None, 130, 32, 256)	1024
conv2d_1 (Conv2D)	(None, 65, 16, 256)	1638656
batch_normalization_1 (BatchNormalization)	(None, 65, 16, 256)	1024
conv2d_transpose (Conv2DTranspose)	(None, 130, 32, 256)	1638656
batch_normalization_2 (BatchNormalization)	(None, 130, 32, 256)	1024
conv2d_transpose_2 (Conv2DTranspose)	(None, 260, 64, 256)	1638656
batch_normalization_3 (BatchNormalization)	(None, 260, 64, 256)	1024
conv2d_transpose_3 (Conv2DTranspose)	(None, 260, 64, 3)	771
cropping2d (Cropping2D)	(None, 257, 62, 3)	0
Total params: 4,940,291		
Trainable params: 4,938,243		
Non-trainable params: 2,048		

Table 2: Model network architecture of proposed CNN-DAE.

seen in Table 5. The achieved PESQ and STOI scores are 1.54 and 0.73, respectively. Although the PESQ score is not as high as one would hope, with previous literature reporting results around 2 (Shivakumar and Georgiou, 2016), the STOI score can be considered to be comparable and even supersedes in some cases results of other baseline algorithms used for speech enhancement, for example in the paper Leglaive et al. (2020). As compared to our own baseline (noisy signals), it is clear that CNN-DAE works well, with both scores improving when using the model.

Moving on to the BBS-Eval Scores, namely SDR, SI-SDR, SAR and ISR, the attained vales are 3.79, 4.23, 7.12, 6.22 dB respectively, greater than the scores of the noisy data by 3.34, 3.59, 5.93 and 1.12 dB respectively. This is an indication that the model performs well in enhancing speech and in predicting signals with reduced number of artefacts, distortion and high overall quality. Finally, observing the achieved results of non-intrusive measures namely, MOSNet and SRMR, it is clear that speech quality based on the perception of people and modulation energy respectively, is improved when compared to the noisy speech signals. Using the CNN-DAE, MOSNet and SRMR scores increase to 2.95 and 6.84, a promising result.

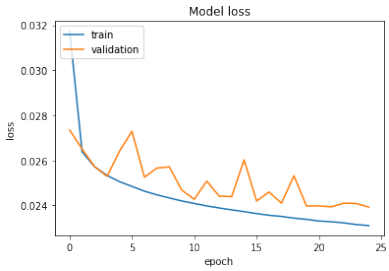


Figure 6: Training and validation loss of CNN-DAE

6.3 Conditional Generative Network (CGAN)

6.3.1 Model Setup

In this case, two architectures need to be designed, the one for the generator and the one for the discriminator. After some experimentation it has been observed that for the generator, increasing the number of filters as the depth of the model increases, is beneficial. The generator, in turn, will be composed by an encoder and a decoder. The decoder will just be a mirroring of the encoder but with two dimensional convolutional transpose layers instead of convolutional ones. The use of convolutional two dimensional layers is due to the dimension of the input which is three dimensional. Hence, the encoder will be composed by four convolutional layers of filters size 8, 16, 32 and 64, respectively. The kernel size of these layers is set to 3×3 as they provided a better and more stable approach than other choices of this parameter. Furthermore, on each layer, a stride of size (2,2) is preferred to maxpooling as they provide a more stable model during training. Each convolution is followed by a non-linear

activation function, parametric rectified linear unit (PReLU). Choosing PReLU as the activation function might be useful in this case since the coefficient of leakage is trained along with the other neural networks parameters, hence it will be more probable to reach the optimal one. The use of only four layers in the decoder is in view of the high training losses given by increasing the number of layers. As stated above, the decoder will just be the "deconvolution" of the encoder, so it will be constructed in a similar way. The only difference is the addition of an extra two dimensional convolutional transpose layer with number of filters 3, kernel size 16×16, and strides $(1, 1)$ in order to accommodate to the dimension of our input. This layer is followed by a PReLU layer and cropping layer in order to further satisfy the desired output shape.

Following a similar architecture for the discriminator as the one for the encoder, has helped in improving the model performance. However, there are some differences as for instance, we want a binary output and the input has 6 channels instead of 3, as now the spectrograms will be conditioned on the noisy sample. Hence, the model will be composed of five convolutional 2D layers and a dense layer applied after flattening the output of the last convolution. The first four convolutional layers will be of the same kernel size and stride size as for the generator case, so 3×3 and $(2, 2)$, respectively. In a similar way, the filter size for each layer will be 8, 16, 32 and 64, respectively. Nevertheless, in this case each convolutional layer will be followed by batch normalisation layer and a Leaky Rectified Linear Unit (Leaky ReLU). This activation function prevents negative values from having a value zero. Then a two dimensional one filter convolutional layer is applied of kernel size 1×1 and stride size $(2, 2)$. The latter is used to reduce the number of parameter in the last dense layer. Furthermore, the output is flattened and then a dense layer is added. The architectures of both the generator and the discriminator are presented in Table 3 and Table 4, respectively.

Layer (type)	Output Shape	Parameters
conv2d_9 (Conv2D)	(None, 128, 30, 8)	224
p_re_lu_9 (PReLU)	(None, 128, 30, 8)	30720
conv2d_10 (Conv2D)	(None, 63, 14, 16)	1168
p_re_lu_10 (PReLU)	(None, 63, 14, 16)	14112
conv2d_11 (Conv2D)	(None, 31, 6, 32)	4640
p_re_lu_11 (PReLU)	(None, 31, 6, 32)	5952
conv2d_12 (Conv2D)	(None, 15, 2, 64)	18496
p_re_lu_12 (PReLU)	(None, 15, 2, 64)	1920
conv2d_transpose_5 (Conv2DTranspose)	(None, 31, 5, 64)	36928
p_re_lu_13 (PReLU)	(None, 31, 5, 64)	9920
conv2d_transpose_6 (Conv2DTranspose)	(None, 63, 11, 32)	18464
p_re_lu_14 (PReLU)	(None, 63, 11, 32)	22176
conv2d_transpose_7 (Conv2DTranspose)	(None, 127, 23, 16)	4624
p_re_lu_15 (PReLU)	(None, 127, 23, 16)	46736
conv2d_transpose_8 (Conv2DTranspose)	(None, 255, 47, 8)	1160
p_re_lu_16 (PReLU)	(None, 255, 47, 8)	95880
conv2d_transpose_9 (Conv2DTranspose)	(None, 270, 62, 3)	6147
p_re_lu_17 (PReLU)	(None, 270, 62, 3)	50220
cropping2d_1 (Cropping2D)	(None, 257, 62, 3)	0
Total params: 369,487		
Trainable params: 369,487		
Non-trainable params: 0		

Table 3: Model network architecture of proposed generator of the CGAN model.

Layer (type)	Output Shape	Parameters
conv2d_13 (Conv2D)	(None, 128, 30, 8)	440
batch_normalization_4 (BatchNormalization)	(None, 128, 30, 8)	32
leaky_re_lu_4 (LeakyReLU)	(None, 128, 30, 8)	0
conv2d_14 (Conv2D)	(None, 63, 14, 16)	1168
batch_normalization_5 (BatchNormalization)	(None, 63, 14, 16)	64
leaky_re_lu_5 (LeakyReLU)	(None, 63, 14, 16)	0
conv2d_15 (Conv2D)	(None, 31, 6, 32)	4640
batch_normalization_6 (BatchNormalization)	(None, 31, 6, 32)	128
leaky_re_lu_6 (LeakyReLU)	(None, 31, 6, 32)	0
conv2d_16 (Conv2D)	(None, 15, 2, 64)	18496
batch_normalization_7 (BatchNormalization)	(None, 15, 2, 64)	256
leaky_re_lu_7 (LeakyReLU)	(None, 15, 2, 64)	0
conv2d_17 (Conv2D)	(None, 8, 1, 1)	65
flatten_1 (Flatten)	(None, 8)	0
dense_1 (Dense)	(None, 1)	9
Total params: 25,298		
Trainable params: 25,058		
Non-trainable params: 240		

Table 4: Model network architecture of proposed discriminator of the CGAN model.

The training is implemented for 100 epochs to enhance the learning, however a stopping point is set for the validation loss in order to avoid over-fitting. The value chosen is when the validation MSE is less than 0.045. This is reasonable since it was observed during training that the MSE does not go under 0.04. Furthermore, the loss for training the generator and discriminator is chosen to be binary cross entropy, which is reasonable given that we are using a GAN model. The optimiser for this training loss is chosen to be Adam since it is able to outperform other optimisers like AdaGrad, RMS Prop, SGD, Nesterov and AdaDelta (Klooster, 2021; Metz et al., 2018). Lastly, the batch size is 16 and the learning rates for the generator and discriminator are chosen to be 0.0003 and 0.003, respectively. The generator has a small learning rate in order to not lose any important features. On the other hand, we want the discriminator to learn as fast as possible to discriminate the spectrograms, hence its learning rate will be higher.

6.3.2 Results

The results are presented in Figure 7 and Figure 8 below. In Figure 7 the generator and discriminator loss in each epoch is displayed. A fast decay be observed in the generator's loss when the number of epochs is small, which then slowly stabilises for around 25 epochs. This implies that the generator is learning and the right choice was made for setting a stopping point since the loss does not give signs of declining significantly much further. On the other hand, the discriminator's loss seem to be very stable for all the epochs. However, there is a small increase as the number of epochs increase, which results in the discriminator being fooled by the generator. Moreover, the validation losses for the binary cross entropy and for the MSE can be seen in Figure 8. Although the validation losses seem to stay quite constant during training, the MSE remains between 0 and 0.2 and the binary cross entropy between 0.7 and 0.8, with irregular large peaks being observed for some epochs. However, these peaks may be explained by the very small scale of the y axis.

Despite having observed good results in the training process, everything seems to change when evaluating it on the test data with the metrics described in Section 5. The PESQ and STOI scores, which are two popular metrics employed in the speech enhancement community, have results of 1.42 and 0.66, respectively. Although, these scores surpassed the baseline of the noisy signals, the model does not seem to obtain the expected results as the PESQ and STOI maximum scores are 4.50 and 1, respectively. Previous studies Pascual et al. (2017) have reported PESQ scores of around 2.16, however it is important to keep in mind that their results for their noisy baseline model where 1.97, which is much higher than our case and while they only obtained a 0.19 dB improvement, we obtained a 0.24 dB increase. Similarly, when comparing the models in Table 5, the model gives decent results by surpassing the noisy signals including the BBS-Eval, the SRMR, and the MOSNet scores. The latter measures people's perception, and can give better ground truth results than the other metrics. Hence, this CGAN model seems to give promising results, but further research and improvements need to be made.

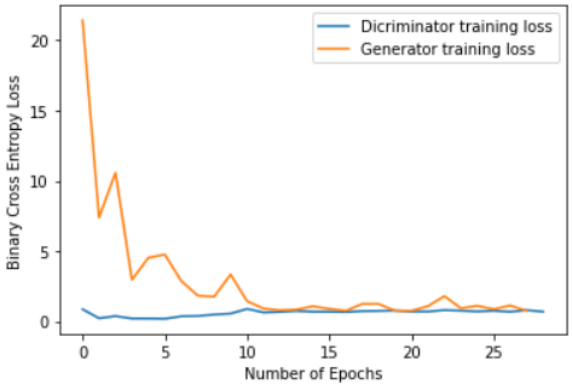


Figure 7: CGANs Binary Cross Entropy Training and Validation Loss

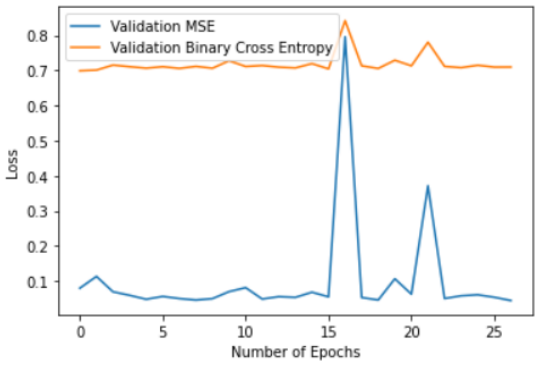


Figure 8: CGANs MSE and Binary Cross Entropy Validation Loss

6.4 Comparison of chosen models

The summary of the results of the models as compared to the original noisy signals can be seen in Table 5. Based on the findings, it can be seen that all models have the effect of improving the speech quality of the signals and reducing noise distortion, although there is a clear winner as to which model achieves the best outcomes. The CNN-DAE produces the most promising results, based on the fact that all metrics significantly increase as compared to baseline values. The second highest performing model is the CGAN, followed by the CRNN, in which there was an improvement to the baseline scores, however not a very significant one, indicating that perhaps the other models are more suited for the problem formulation and this particular task.

Having identified the best model as the CNN-DAE, further evaluation was conducted in terms of our own visual and auditory perception of its predicted signals. Given the fact that "human subjective evaluation is the "gold standard" to evaluate speech quality" (Reddy et al., 2021), a sample of predicted signals were converted back to wav files, to enable one to hear them and compare them directly with their clean and noisy counterparts. Having listened to the predicted signals (please refer to the the "CNN-DAE & CGANS & REVERSE" notebook), one can clearly identify the spoken words and the noises have completely dissipated. There is however a slight ringing sound introduced, which may account for a lower PESQ score, as compared to other speech enhancement models. The oscillogram and spectrogram of samples of predicted, noisy and clean speech can be visualised below in Figure 9, Figure 10, Figure 11 and Figure 12.

Metric	Noisy Signals	CNN-DAE	CGAN	CRNN
SDR Score (dB)	0.45	3.79	1.46	0.90
SI-SDR Score (dB)	0.64	4.23	1.78	1.14
SAR Score (dB)	1.19	7.12	4.39	3.47
ISR Score (dB)	5.10	6.22	5.17	5.26
PESQ Score	1.18	1.54	1.42	1.40
STOI Score	0.42	0.73	0.66	0.64
MOSNET Score	2.47	2.95	2.68	2.51
SRMR Score	4.39	6.84	6.22	6.08

Table 5: Metrics for noisy baseline, CNN-DAE, CGAN, and CRNN.

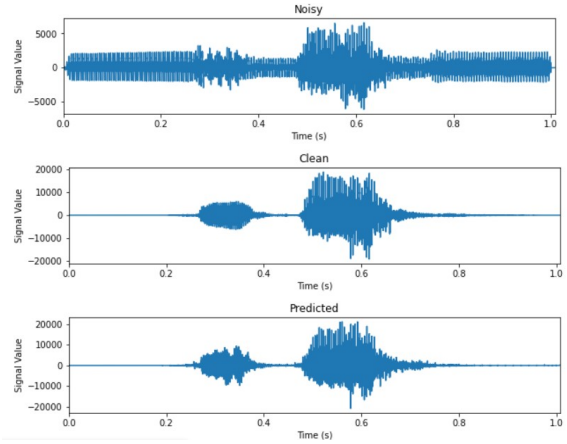


Figure 9: Oscillograms of Predicted, Noisy and Clean Sample of CNN-DAE

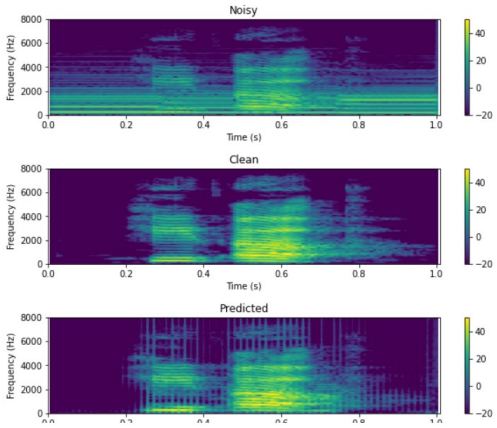


Figure 10: Spectrograms of Predicted, Noisy and Clean Sample of CNN-DAE

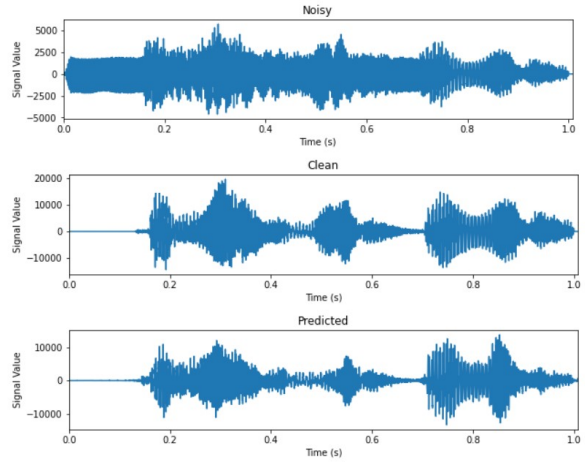


Figure 11: Oscillograms of Predicted, Noisy and Clean Sample of CNN-DAE

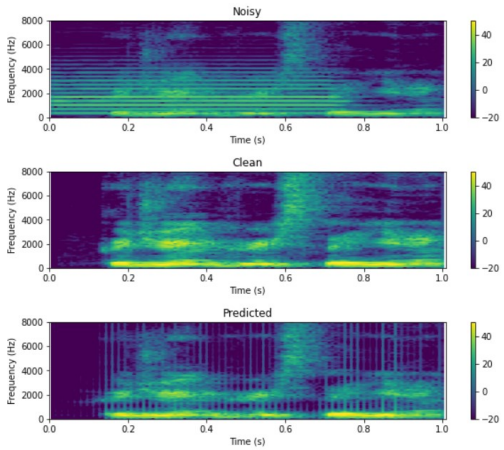


Figure 12: spectrograms of Predicted, Noisy and Clean Sample of CNN-DAE

7 Reverse Model

For the final stage of our investigation we introduce the novel approach of training a neural network to add instead of remove noise, as explained in Section 1. These could serve as a valuable stepping stone into further future research of this topic. The methodology implemented and achieved results are described below.

7.1 Implementation and Results

Given the success of the Convolutional Denoising Auto Encoder, in noise suppression when trained on noisy inputs and clean targets, we decided to investigate its performance if the roles were reversed. In particular, we trained the same structure of model described in Section 6.2.1, for 20 epochs, with input the clean speech data and target the noisy data. First looking at Figure 13, it is clear that the model improves in producing signals which are comparable to the noisy signals, with the training and validation losses both decreasing to 0.0306 and 0.0311 respectively. This loss is greater than the training and validation loss of the original CNN-DAE (Section 6.2.2), indicating that the structure of the auto encoder learns to predict clean speech better than it learns to predict noisy speech. Evaluating the reverse model on the test signals, gave an MSE of 0.0314, which although is low, it's an indication that there is still some room for improvement.

To have a visual perception of the predicted signals, the spectrograms and oscillograms of a random sample are depicted below in Figure 14 and Figure 15. These clearly indicate that although the predicted noisy speech does not exactly follow the shape of the true noisy speech, noise is added in the correct areas and is thus a result that has potential. To now have an audible indication of the performance, samples of predicted signals can be listened to in the "CNN-DAE_CGANS_REVERSE" notebook. These primarily include the spoken words combined with

higher frequency noises, indicating that the model has learnt to add noise to the clean speech files. The noise added however cannot be discerned to be anything specific, for example applause or bells, indicating that further training may be required on specific types of noises. This would hopefully enable the model to learn the specific frequency and intensity patterns followed by each noise, thus training it to add clear and correct formulations of noises, to the clean speech data. Given the fact that our model has been trained on hundreds of different noises, it is natural for the model not to be able to recreate a specific noise, however it serves as a valuable base model onto which further research can be conducted, to investigate the concept of deep learning audio speech addition.

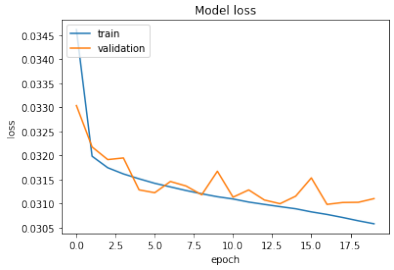


Figure 13: Training and validation loss of Reverse CNN-DAE

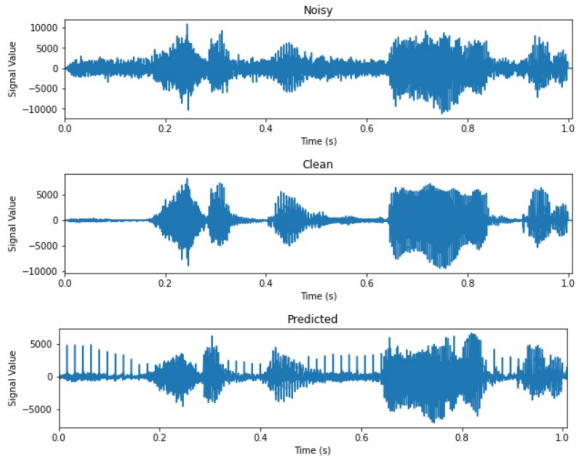


Figure 14: Oscillograms of Predicted, Noisy and Clean Sample of Reverse CNN-DAE

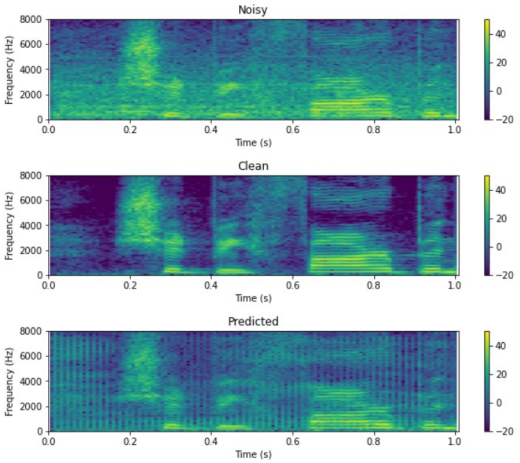


Figure 15: Spectrograms of Predicted, Noisy and Clean Sample of Reverse CNN-DAE

8 Conclusion

To summarise, in this paper, three different neural network architectures applied to noise suppression were taken under consideration. Specifically, the Convolutional Recurrent Neural Network (CRNN), the Convolutional Denoising Auto Encoder (CNN-DAE) and the Conditional Generative model (CGAN) were studied. Although all the methods outperform the baseline results, the CNN-DAE clearly surpassed the other two. Nevertheless, the results are far from being optimal even after applying feature data engineering, data generation and intensive training. Hence, none of the models seem reasonable for the speech enhancement application and a different approach with these models may need to be followed.

Furthermore, a reverse application of speech enhancement is explored. Although seemingly counter intuitive, adding noise to clean speech could prove to be useful in future applications such as sound effects in the entertainment sector, in particular the film, gaming and television industries. To tackle this unorthodox but nevertheless lucrative and intriguing topic, we decided to use the model which showed the greatest potential, the CNN-DAE. Although the generated results were audio files consisting of a combination of many noises indistinguishable from one another, a natural result based on the vast variety of noises used in training, a baseline model was designed for further exploration. Given the ultimate goal is to build a neural network capable of leaning to add specific noises to the data (taking into account the amplitude of frequencies of the clean noise data so as to not overshadow it), future research could focus on the implementation of data-sets with a specific type of noisy and clean input, for example people talking in a cafe environment.

Moving forward, it would also be interesting to explore other type of recurrent neural networks for the CRNN

model like GRU, whose simpler structure may be preferred to avoid overfitting the training data. Moreover, building a stacked deep denoising autoencoder might improve the performance for the CNN-DAE even further. In such a formulation, each layer is trained first in an unsupervised manner using the encoded input of the previous layer, and this is then followed by fine tuning of the whole model in a supervised manner. Such models have been seen to operate well in similar applications and thus may be considered in future investigations. When addressing the CGAN, the usage of skipping connections and the addition of regularisation terms in the generator loss may lead to an improvement in their performance as described by previous literature, and thus such possibilities may also be explored further. Finally, the usage of even more types of noises and the potential creation of an ensemble model is suggested, to further increase the generalisation capacity of each of the models, as well as combine the positive aspects of the three, to construct and hopefully achieve the ultimate noise suppression model.

References

Agarap, A. F. (2018, 03). Deep learning using rectified linear units (relu).

Alain, G. and Y. Bengio (2014, jan). What regularized auto-encoders learn from the data-generating distribution. *J. Mach. Learn. Res. 15*(1), 3563–3593.

Allen, J. and L. Rabiner (1977). A unified approach to short-time fourier analysis and synthesis. *Proceedings of the IEEE 65*(11), 1558–1564.

Arjovsky, M. and L. Bottou (2017). Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*.

Baby, D. (2020). isegan: Improved speech enhancement generative adversarial networks. *arXiv preprint arXiv:2002.08796*.

Bajaj, K., D. K. Singh, and M. A. Ansari (2020). Autoencoders based deep learner for image denoising. *Procedia Computer Science 171*, 1535–1541.

Basant Agarwal, Valentina Emilia Balas, L. C. J. R. C. P. M. (2020). *Deep Learning Techniques for Biomedical and Health Informatics*.

Beerends, J., A. Hekstra, A. Rix, and M. Hollier (2002, 10). Perceptual evaluation of speech quality (pesq) - the new itu standard for end-to-end speech quality assessment - part ii - psychoacoustic model. *Journal of the Audio Engineering Society. Audio Engineering Society 50*.

Bengio, Y., A. Courville, and P. Vincent (2013, 08). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence 35*, 1798–1828.

Bengio, Y., P. Simard, and P. Frasconi (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks 5*(2), 157–166.

Blackman, R. B. and J. W. Tukey (1958). The measurement of power spectra from the point of view of communications engineering — part i. *The Bell System Technical Journal 37*(1), 185–282.

Brownlee, J. (2018). What is the difference between a batch and an epoch in a neural network? *Machine Learning Mastery 20*.

Cakır, E., G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen (2017). Convolutional recurrent neural networks for polyphonic sound event detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing 25*(6), 1291–1303.

Chen, J., J. Benesty, Y. Huang, and S. Doclo (2006). New insights into the noise reduction wiener filter. *IEEE Transactions on audio, speech, and language processing 14*(4), 1218–1234.

Cho, K., B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Choi, K., G. Fazekas, M. Sandler, and K. Cho (2017). Convolutional recurrent neural networks for music classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2392–2396. IEEE.

Creswell, A., K. Arulkumaran, and A. A. Bharath (2017). On denoising autoencoders trained to minimise binary cross-entropy. *arXiv preprint arXiv:1708.08487*.

Donahue, C., B. Li, and R. Prabhavalkar (2018). Exploring speech enhancement with generative adversarial networks for robust speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5024–5028. IEEE.

Dumoulin, V. and F. Visin (2016). A guide to convolution arithmetic for deep learning.

Elman, J. L. (1990). Finding structure in time. *Cognitive science 14*(2), 179–211.

Emmert-Streib, F., Z. Yang, H. Feng, S. Tripathi, and M. Dehmer (2020, 02). An introductory review of deep learning for prediction models with big data. *Frontiers in Artificial Intelligence 3*, 4.

Falk, T. H. and W.-Y. G. Chan (2008). A non-intrusive quality measure of dereverberated speech.

Ferguson, S. H. and D. Kewley-Port (2007). Talker differences in clear and conversational speech: Acoustic characteristics of vowels.

Fu, S.-W., Y. Tsao, and X. lu (2016, 09). Snr-aware convolutional neural network modeling for speech enhancement. pp. 3768–3772.

Févotte, C., R. Gribonval, and E. Vincent (2005, 01). $\mathrm{Bss}_e valtoolboxuserguide - -revision2.0$.

Gondara, L. (2016). Medical image denoising using convolutional denoising autoencoders. In *2016 IEEE 16th international conference on data mining workshops (ICDMW)*, pp. 241–246. IEEE.

Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Graves, A., A.-r. Mohamed, and G. Hinton (2013). Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 6645–6649. Ieee.

Hall, M. G., A. V. Oppenheim, and A. S. Willsky (1983). Time-varying parametric modeling of speech. *Signal Processing 5*(3), 267–285.

Harris, F. J. (1978). On the use of windows for harmonic analysis with the discrete fourier transform. *Proceedings of the IEEE 66*(1), 51–83.

He, K., X. Zhang, S. Ren, and J. Sun (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778.

Hitaj, B., G. Ateniese, and F. Perez-Cruz (2017). Deep models under the gan: information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pp. 603–618.

Hochreiter, S. and J. Schmidhuber (1997). Long short-term memory. *Neural computation 9*(8), 1735–1780.

Iandola, F., S. Han, M. Moskewicz, K. Ashraf, W. Dally, and K. Keutzer (2016, 02). Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡0.5mb model size.

Isola, P., J.-Y. Zhu, T. Zhou, and A. A. Efros (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125–1134.

Ivry, A., I. Cohen, and B. Berdugo (2021). Objective metrics to evaluate residual-echo suppression during double-talk. *CoRR abs/2107.07471*.

Karsoliya, S. (2012). Approximating number of hidden layer neurons in multiple hidden layer bpnn architecture. *International Journal of Engineering Trends and Technology 3*(6), 714–717.

Kingma, D. and J. Ba (2014a, 12). Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Kingma, D. P. and J. Ba (2014b). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Klooster, L. (2021). Approximating differential equations using neural odes. B.S. thesis, University of Twente.

Kluender, K. R., C. E. Stilp, and M. Kiefte (2013). Perception of vowel sounds within a biologically realistic model of efficient coding. In *Vowel inherent spectral change*, pp. 117–151. Springer.

Kovenko, V. and I. Bogach (2020). A comprehensive study of autoencoders' applications related to images. In *IT&I Workshops*.

Le Roux, J., S. Wisdom, H. Erdogan, and J. Hershey (2019, 05). Sdr – half-baked or well done? pp. 626–630.

Leglaive, S., X. Alameda-Pineda, L. Girin, and R. Horaud (2020). A recurrent variational autoencoder for speech enhancement. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 371–375.

Li, K., J. Daniels, C. Liu, P. Herrero, and P. Georgiou (2019). Convolutional recurrent neural networks for glucose prediction. *IEEE journal of biomedical and health informatics 24*(2), 603–613.

Li, Y., R. Yu, C. Shahabi, and Y. Liu (2017). Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*.

Liebl, B. and M. Burghardt (2020). On the accuracy of crnns for line-based ocr: A multi-parameter evaluation. *arXiv preprint arXiv:2008.02777*.

Lightburn, L. and M. Brookes (2016). A weighted stoi intelligibility metric based on mutual information. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5365–5369.

Lo, C.-C., S.-W. Fu, W.-C. Huang, X. Wang, J. Yamagishi, Y. Tsao, and H.-m. Wang (2019, 09). Mosnet: Deep learning-based objective assessment for voice conversion. pp. 1541–1545.

Lu, X., Y. Tsao, S. Matsuda, and C. Hori (2013). Speech enhancement based on deep denoising autoencoder. In *Interspeech*, Volume 2013, pp. 436–440.

Marco Leo, G. M. F. (2018). *Computer Vision for Assistive Healthcare*.

Masters, D. and C. Luschi (2018, 04). Revisiting small batch training for deep neural networks.

Metz, L., N. Maheswaranathan, J. Nixon, D. Freeman, and J. Sohl-Dickstein (2018). Learned optimizers that outperform sgd on wall-clock and test loss. *arXiv preprint arXiv:1810.10180*.

Michelsanti, D. and Z.-H. Tan (2017). Conditional generative adversarial networks for speech enhancement and noise-robust speaker verification. *arXiv preprint arXiv:1709.01703*.

Mou, J. and J. Li (2021, 02). Effects of number of filters of convolutional layers on speech recognition model accuracy.

Nwankpa, C., W. Ijomah, A. Gachagan, and S. Marshall (2020, 12). Activation functions: Comparison of trends in practice and research for deep learning.

Park, Y. (2022). Concise logarithmic loss function for robust training of anomaly detection model. *arXiv preprint arXiv:2201.05748v1*.

Pascual, S., A. Bonafonte, and J. Serra (2017). Segan: Speech enhancement generative adversarial network. *arXiv preprint arXiv:1703.09452*.

Pielawski, N. and C. Wählby (2020). Introducing hann windows for reducing edge-effects in patch-based image segmentation. *PloS one 15*(3), e0229839.

QingJie, W. and W. WenBin (2017). Research on image retrieval using deep convolutional neural network combining l1 regularization and prelu activation function. In *IOP Conference Series: Earth and Environmental Science*, Volume 69, pp. 012156. IOP Publishing.

Reddy, C. K. A., V. Gopal, and R. Cutler (2021). Dnsmos p.835: A non-intrusive perceptual objective speech quality metric to evaluate noise suppressors. *CoRR abs/2110.01763*.

Riad, R., O. Teboul, D. Grangier, and N. Zeghidour (2022). Learning strides in convolutional neural networks. *CoRR abs/2202.01653*.

Rix, A., J. Beerends, M. Hollier, and A. Hekstra (2001). Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs. In *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*, Volume 2, pp. 749–752 vol.2.

Salman, A. G., Y. Heryadi, E. Abdurahman, and W. Suparta (2018). Single layer & multi-layer long short-term memory (lstm) model with intermediate variables for weather forecasting. *Procedia Computer Science 135*, 89–98.

Santos, J., M. Senoussaoui, and T. Falk (2014, 09). An improved non-intrusive intelligibility metric for noisy and reverberant speech. pp. 55–59.

Schroeder, M. R. (2004). *Computer speech: recognition, compression, synthesis*, Volume 35. Springer Science & Business Media.

Sergey Ioffe, C. S. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *JMLR: WCP volume 37*.

Shivakumar, P. G. and P. G. Georgiou (2016). Perception optimized deep denoising autoencoders for speech enhancement. In *INTERSPEECH*.

Siddharth Sharma, Simone Sharma, A. A. (2020). Activation functions in neural networks. *International Journal of Engineering Applied Sciences and Technology, 2020 Vol. 4, Issue 12, ISSN No. 2455-2143, Pages 310-316*.

Sonnaillon, M. O. and F. J. Bonetto (2005). A low-cost, high-performance, digital signal processor-based lock-in amplifier capable of measuring multiple frequency sweeps simultaneously. *Review of Scientific Instruments 76*(2), 024703.

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research 15*(56), 1929–1958.

Storey, B. D. (2002). Computing fourier series and power spectrum with matlab. *TEX paper 660*, 661.

Taal, C. H., R. C. Hendriks, R. Heusdens, and J. Jensen (2011). An algorithm for intelligibility prediction of time–frequency weighted noisy speech. *IEEE Transactions on Audio, Speech, and Language Processing 19*(7), 2125–2136.

Tan, K. and D. Wang (2018). A convolutional recurrent neural network for real-time speech enhancement. In *Interspeech*, Volume 2018, pp. 3229–3233.

Tang, D., B. Qin, and T. Liu (2015). Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pp. 1422–1432.

Thakur, B. and R. Mehra (2016). Discrete fourier transform analysis with different window techniques algorithm. *International Journal of Computer Applications 975*, 8887.

Venkataramani, S., R. Higa, and P. Smaragdis (2018, 11). Performance based cost functions for end-to-end speech separation. pp. 350–355.

Vetterli, M., P. Marziliano, and T. Blu (2002). Sampling signals with finite rate of innovation. *IEEE transactions on Signal Processing 50*(6), 1417–1428.

Vincent, E., R. Gribonval, and C. Fevotte (2006). Performance measurement in blind audio source separation. *IEEE Transactions on Audio, Speech, and Language Processing 14*(4), 1462–1469.

Vincent, E., H. Sawada, P. Bofill, S. Makino, and J. Rosca (2007, 09). First stereo audio source separation evaluation campaign: data, algorithms and results.

Vincent, P., H. Larochelle, Y. Bengio, and P.-A. Manzagol (2008, 01). Extracting and composing robust features with denoising autoencoders. pp. 1096–1103.

Wang, R., Z. Li, J. Cao, T. Chen, and L. Wang (2019). Convolutional recurrent neural networks for text classification. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–6. IEEE.

Wang, Y., Y. Li, Y. Song, and X. Rong (2020, 03). The influence of the activation function in a convolution neural network model of facial expression recognition. *Applied Sciences 10*, 1897.

Wilson, D. and T. Martinez (2001, 02). The need for small learning rates on large problems. Volume 1, pp. 115 – 119 vol.1.

Wu, H. and S. Prasad (2017). Convolutional recurrent neural networks forhyperspectral data classification. *Remote Sensing 9*(3), 298.

Xu, J., Z. Li, B. Du, M. Zhang, and J. Liu (2020). Reluplex made more practical: Leaky relu. In *2020 IEEE Symposium on Computers and Communications (ISCC)*, pp. 1–7. IEEE.

Xu, Y., J. Du, L.-R. Dai, and C.-H. Lee (2014). A regression approach to speech enhancement based on deep neural networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing 23*(1), 7–19.

Zia, T. and U. Zahid (2019). Long short-term memory recurrent neural network architectures for urdu acoustic modeling. *International Journal of Speech Technology 22*(1), 21–30.

9 Statement of Title

Each of the candidates focused on one of the three models. For the rest of the project, all of the candidates contributed equally.